APPLICATION FOR PATENT

of

Michel Shane Simpson
Brett Dee Garrett
Nathan Blaine Jensen
and
William Donald Peterson III

for

METHOD AND APPARATUS FOR PRESENTING, SEARCHING, AND VIEWING DIRECTORIES

Graydon Head & Ritchey LLP
1900 Fifth Third Center
511 Walnut Street
Cincinnati, Ohio 45202
(513) 621-6464

COPYRIGHTS PRESERVED

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

TECHNICAL FIELD

The present invention relates generally computer systems and software, and will be specifically disclosed as a method and apparatus for presenting, searching, and viewing directories.

BACKGROUND

The virtual explosion of technical advances in microelectronics, digital computers and software have changed the face of modem society. In fact, these technological advances have become so important and pervasive that this explosion is sometimes referred to as "the information revolution." Through telephone lines, networks, satellite communications and the like, information and resources are ever increasingly being accessed and shared.

One mechanism used to organize, store, search, edit, and retrieve information and resources on a computer is through a directory. A directory contains a collection of objects, sometimes referred to as identities, with associated attributes or properties. For example, an directory object could be a User object that represents a human user. Beyond representing users, objects can represent things that humans relate to when dealing with computers. For instance, some typical objects might represent printers, print queues, files, resources, computers, and the like. In addition, objects can represent non-computer related things such as countries, companies, organizations, departments, buildings, and the like. Furthermore, objects can be organizational in nature to group other objects together. As one with ordinary skill in the art will readily appreciate, objects can represent virtually anything, whether imaginary or real.

A directory object has a variety of associated attributes. For instance, a user object may have attributes such as Given Name, Last Name, Title, Location, Telephone Number, Description, Authentication, Login Name, etc. In most cases, an

object attribute will have associated attribute data. For example, the value for the attribute "Given Name" might be "George". An attribute is usually based on an attribute type, which has an attribute syntax. The data which can be entered as a value associated with the attribute is dictated by the attribute syntax. For instance, some typical attribute types include Back Link, Boolean, Case Exact String, Case Ignore List, Case Ignore String, Class Name, Counter, Distinguished Name, E-mail Address, Facsimile Telephone Number, Hold, Integer, Interval, Net Address, Numeric String, Object ACL, Octet List, Octet String, Path, Postal Address, Printable String, Replica Pointer, Stream, Telephone Number, Time, Timestamp, Typed Name, and Unknown. Each of these attribute types has a predetermined attribute syntax or data structure appropriate for the type of data that will be entered as a value.

In many cases, the structure of a directory is governed by a schema. The schema defines the rules for adding and managing objects and object attributes in a directory. For instance, a schema can be specified through a data dictionary that provides a standard set of data types or classes from which objects can be created. In such case, each object in the directory belongs to a class that specifies which attributes are associated with the object. In other words, an object is of a particular type and is created from a corresponding class. For example, a User object would be created from a User class, a Printer object would be created from a Printer class, etc. An object created from a class is sometimes referred to as being "instantiated" from the class, wherein the object is referred to as an "instance" of the class. Generally, such schemas are extensible so that it may be tailored to modify existing classes or add new classes.

Beyond controlling the structure of the directory objects, the schema may also control the relationship among the objects in the directory. In controlling this relationship, the schema may specify subordination among object classes. For instance, for every object there can be a group of classes from which subordinate objects can be formed. Objects that contain other objects are called container objects, which are the building blocks of the hierarchal relationship of objects in the directory. Objects that cannot contain other objects are known as non-container or leaf objects. As such, the combination of container and leaf objects can be arranged in a tree-like structure representing the hierarchal relationship of the various objects.

Directories that include such schemas are sometime referred to as hierarchal directories.

There are a variety of types, formats and standards for directories. For example, some directories are compliant with the Lightweight Directory Access Protocol ("LDAP"). LDAP is a protocol used to access a directory. One common use for LDAP compliant directories is in conjunction with Web browsers and e-mail programs, however, LDAP directories are used in a variety of other applications. Another example is X.500 directories, which are hierarchal directories that use the Directory Access Protocol ("DAP") protocol to access such directories. Beyond these two limited examples, a variety of other types of directories are currently known to one with ordinary skill in the art. Moreover, additional types of directories will undoubtably emerge in the future. Some current examples of directories include Novell Directory Services "NDS", Novell's eDirectory, Oracle Databases, Netscape's Directory Server and Planet Directory Server, Microsoft's Active Directory, IBM's SecureWay Directory.

Sometimes, the directories on various computers are coordinated to provide a larger directory, which is sometimes referred to as a distributed directory. Generally speaking, a distributed directory spans and is shared by multiple networked servers wherein the objects in the directory are synchronized between the various servers, although a single server can also maintain a distributed directory. While distributed directories are often used with client/server networks, they are not necessarily limited to the context of such networks. Information on the distributed directory can be created, read, modified, and shared by other networked nodes, such as client computers or other servers, assuming such nodes have appropriate access rights to the distributed directory.

SUMMARY

One aspect of the present invention is a computer system. One or more directories are accessible by a computer. The directories comprise a plurality of directory objects wherein each object has object attributes and attribute data. A directory shell references the directories and has one or more categories. Each of the categories is associated with one or more directory objects and at least a portion of the associated object attributes. A user interface is operable for a user to query the categories in the directory shell. A directory interface is operable to send a

search request to the directories in accordance with the query and receive attribute data from the directories satisfying the search request.

Another aspect of the present invention is a method in a computer system. A directory shell is created comprising one or more categories. The directory shell is associated with one or more directories wherein each directory comprises a plurality of directory objects having object attributes and attribute data. Each of the categories is associated with one or more directory objects and at least a portion of the object attributes corresponding to the directory objects. A search is requested for query data against a selected category. The directories are searched for the query data against the attribute data corresponding to the directory objects and object attributes associated with the selected category. Attribute data satisfying the search is received and presented.

Yet another aspect of the present invention is a data structure. One or more directories each comprise a plurality of classes with attributes and a plurality of objects instantiated from the classes. The instantiated objects comprise data associated with the attributes. A directory shell is associated with the directories. A plurality of categories are associated with the directory shell wherein each category corresponds to one or more classes in the directories. A plurality of category attributes are associated with each category wherein each category attribute corresponds to an attribute of the class of the associated class. The directory shell is queryable against the categories and category attributes to search and retrieve data of the objects in the directories.

Still other aspects of the invention will become apparent to those skilled in the art from the following description, which is by way of illustration, one of the best modes contemplated for carrying out the invention. As will be realized, the invention is capable of other different and obvious aspects, all without departing from the invention. Accordingly, the drawings and descriptions are illustrative in nature and not restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, incorporated in and forming part of the specification, illustrate several examples of the present invention and, together with their descriptions, serve to illustrate aspects of the invention. Like numerals indicate the same element throughout the views. In the drawings:

Fig 1 depicts an example of a computer system, and more particularly the relationship between a directory shell and a plurality of directories;

Fig. 2 illustrates an example of a data structures for a directory and a directory shell;

Fig. 3 illustrates an example of a flow chart of a method for using such data structures;

Fig. 4 depicts an example of a screen shot of the directories option of an administration utility;

Fig. 5 depicts an example of a screen shot of the LDAP settings tab for the directories option of an administration utility;

Fig. 6 depicts an example of a screen shot of an attributes tab for the directories option of an administration utility;

Fig. 7 depicts an example of a screen shot of an advanced tab of the directories option for the directories option of an administration utility;

Fig. 8 depicts an example of a screen shot of a directory browser;

Fig. 9 depicts an example of a screen shot of a directory browser in a modify mode;

Fig. 10 depicts an example of a screen shot of a directory browser which includes an organizational chart; and

Fig. 11 depicts an example of a screen shot of a directory browser with a Boolean query format.

DETAILED DESCRIPTION

Fig. 1 illustrates an example of a computer system of the present invention. As shown in this example, the directories 10a, 10b, and 10c are each maintained by directory servers 12a, 12b, and 12c, respectively. While three directories 10 are illustrated in this example, the invention could also be used with a single directory or with many more than three directories. Further, invention could be used in conjunction with any type of directory, distributed or otherwise, which could be maintained on a single or multiple servers. In addition, the invention is not limited to servers maintaining directories, as any computer or node that maintains or accesses a directory could also be employed. The directory servers 12 in this example communicate with the computer 30. The directory server 12a is directly connected to the computer 30, such as on a local area network "LAN", wide area network

"WAN", peer-to-peer connection, or other ways known in the art, or possibly be one in the same as the computer 30. The directory servers 12b and 12c communicate with the computer 30 over the Internet 20. As one with ordinary skill in the art will readily recognize, the directory servers 12 can communicate with the computer 30
5   using a variety of different mechanisms, formats, and protocols.

The computer 30 can receive data and instructions to operate computer 30 (often referred to as programs or applications) in a variety of different ways. Two examples are illustrated. The computer readable medium 32, shown here as a floppy diskette, holds information readable by a computer 30, such as programs,
10 instructions, data, files, etc. As one with ordinary skill in the art will readily appreciate, computer readable medium can take a variety of forms, including magnetic storage (such as hard disk drives, floppy diskettes, etc.), optical storage (such as laser discs "LD", compact discs "CD", digital versatile discs "DVD", etc.), electronic storage (such as random access memory "RAM", read only memory
15 "ROM", programmable read only memory "PROM", flash memory, etc.), and the like. Certain types of computer readable medium, which are sometimes described as being nonvolatile, can retain data in the absence of power so that the information is available when power is restored. The propogated signal 34 is a frequency transmission that contains data and instructions understandable to the computer 30.
20 Some typical examples of propogated signals include electrical pulses transmitted over telephone lines or network cables, optical signals such as infrared, radio or microwave frequencies, originating from towers or satellites, and the like.

A directory shell application 40 runs on the computer, which, for example, may have been received by the computer 40 on a computer readable medium or
25 propogated signal. The directory shell 40 includes references to the directories 10 and includes one or more categories. Each category is associated with one or more objects in the directories 10. The categories may also be associated with all or a subset of the object attributes in the directories 10. A directory interface 46 formats, submits, and receives data to/from the directories 10. As different directories have
30 different protocols, it is preferred that the directories 10 each be LDAP compliant, however, if a given directory 10 is not LDAP compliant, the directory drivers 34 can provide the necessary format/protocol information so that the directory interface 46 can communicate with such directory. A user interface 44 allows a user to view, search and interact with the directories 10. For instance, the user interface 44 could

allow the user to format a query directed to objects in the directories 10 and present the object data that satisfy the query. Preferably, such queries are formatted against the categories. The user interface 44 could take a variety of forms known as the art,

5   including an application device (e.g. cell phone, personal digital assistant "PDA", and the like), applet, serverlet, function in a library, hypertext makeup language, extensible makeup language, wireless makeup language, dynamic hypertext makeup language, and the like. Further, the user interface 44 can operate on any device receiving wired or wireless data, including workstations, PDA's, cell phones, computers, and the like. The directory interface 46, directory shell 40, and user

10   interface 44 can be an integrated set of instructions, a separate module of instructions, part of a library of instructions, any combination of the foregoing, or in other formats known in the art.

Fig. 2 illustrates an example of data structures for a directory 50 and a directory shell 60. One or more directories 50 each have a plurality of directory

15   objects 53. In the present example, the directory objects 53 are instantiated from directory classes 51. Each directory class 51 has a plurality of class attributes 52. When an directory object 53 is instantiated, the directory object 53 will contain the class attributes 52 corresponding to the directory class 51 from which the directory object 53 was instantiated. Object data 54 populates the class attributes 52 of an

20   instantiated directory object 53. It should be noted that the foreing data structure of a directory 50 is only an example, and other structures are also contemplated. For instance, a directory object 53 could be created independent of a directory class 51, and nevertheless have its own set of attributes.

The directory shell 60 comprises one or more directory references 61. Each

25   directory reference 61 is associated with a directory 50 and can include a variety of data specific to the directory 50. The directory references 61 identify which directories that the directory shell 60 will access. The directory shell 60 also includes a collection of categories 62. Preferably, each directory reference 61 has its own unique set of associated categories 62. A category 62 represents and references

30   one or more directory classes 51 that have been grouped together under one category name. In other words, any given category 62 is a collection of directory classes 51 and the directory objects 53 instantiated from such directory classes 51. A category 62 can reference classes 51 in a single directory or classes of different directories. For instance, a category called "Find People" can be configured to

directory for objects instantiated from a User directory class. In another example, a category called "Cisco People" can be configured to search only the Cisco directory for User objects. In another example, a category called "Find All" can be configured to search both the Novell and Cisco directories for User objects. As one with 5 ordinary skill in the art will recognize, the combinations and flexibility of categories 62 is virtually unlimited.

Each category 62 includes one or more category attributes 63. Each category attribute 63 is associated with or references (sometimes referred to as mapped) a class attribute 52 in the directory 50. However, every class attribute 52 does not 10 need to be mapped to a category attribute 63. Accordingly, the category attributes 63 for a given category 62 are totally configurable to reference any class attribute 52 in the corresponding directory class 51. Optionally, the mapping between category attributes 63 and class attributes 52 can be predefined based on a library of standard categories 62. For instance, a standard "user" category in such a library would 15 automatically map the category attributes to the class attributes Lastname, Firstname, and Organization of the User directory class.

Fig. 3 illustrates a method 70 utilizing the data structures 50 and 60. At step 71, a directory shell 60 is created. At step 72, one or more directories 50 are associated with the directory shell 60. For instance, a directory reference 61 can be 20 created for each directory 50 associated with the directory shell 60. During step 73, a plurality of categories 62 are associated with directory objects 53. For instance, each category 62 could reference one or more directory classes 51 such that directory objects 53 instantiated from such directory classes 51 are associated to the category 62. At step 74, category attributes 63 are associated or mapped to object 25 attributes, such as to the class attributes 52, which are included in an instantiated directory object 53. At step 75, a search request to sent to the directory 50 based on the categories 62 and category attributes 63 for the corresponding directory reference 61. For instance, a query can be formatted such that query data is matched against a selected category and the category attributes. In such an 30 instance, the category may be referred to as a search object. The formatted query could then be translated to search for the query data against the directory objects in the directory that corresponds to the selected category. Function calls using LDAP may be employed to transmit the request to the directory, assuming the directory is LDAP compliant. In another example, function calls could be prepared in accordance

with appropriate directory drivers. The directory 50 replies to the search request and object data 54 satisfying the search is returned to the directory shell 60 during step 76. At step 77, the received object data 54 is presented, preferrable in a user interface associated with the directory shell 60. It should be noted that the sequence
5   of the steps in method 70 are presented as an illustration and the sequence may be varied.

By way of example, the following illustrates a data structure created from Java classes, wherein an instantiated class becomes an object from that class: Each Java class comprises data members and method members.
10

Schemer Class – This is a utility class that dynamically builds an internal table of DirectoryScheme objects that can be referenced by name though its API's.

DirectoryScheme Class – Each instance of this class corresponds to a single
15   directory reference.   This class is a top-level class that contains directory information to which an instance of the class is associated.   This class also contains a table of all ObjectScheme objects for this directory reference.   All contained ObjectScheme objects can be requested by name through API's of a DirectoryScheme object.
20

ObjectScheme Class – Each instance of this class corresponds to a category. This class contains a table of all directory classes associated with this category.   This class also contains a table of all AttributeScheme objects and their association with directory classes that are associated with the
25   ObjectScheme object.

AttributeScheme Class – Each instance of this class corresponds to an category attribute. This class contains a table of all administrator settings and schema data for a class attribute. All data contained in this object is available
30   through API's.

AttributeSchemaReader Class – This is a utility class that builds one AttributeSchemaTableEntry object per class attribute of a given directory.

This class builds a dynamic list of class attribute data that is made available through public API's.

AttributeSchemaTableEntry Class – This class contains schema information in a given class attribute and is intended to be a data member of the AttributeScheme class, where API's expose the contained data.

Naturally, the foregoing Java class structure is illustrative and a variety of other object oriented class structures, as well as other data and method structures, are contemplated. As one with ordinary skill in the art will readily recognize, many alternatives and variations may be employed.

One example a directory shell, which operates in conjunction with the foregoing Java class structure, is used in conjunction with Novell's eGuide product. This example of a directory shell includes two aspects: an administration utility and a directory browser. The administration utility is generally used by system administrators to configuring and managing the directory shell. The directory browser, is typically used by end users to search and view directories, and if appropriate rights are granted, modify the directory. In the present example, the user interface for both aspects of the directory shell are generated by serverlets and are presented in hypertext markup language ("HTML") and viewed in a standard browser, however, a variety of alternative user interfaces and variations are also contemplated, such as stand alone applications, applets, wireless devices, and the like.

Fig. 4 illustrates a page 100 associated with an administration utility. As shown in this example, the page 100 generally comprises two portions: an options panel 110 and a details panel 120. The options panel 110 comprises a list of a variety of options 112. In the present example, the options 112 are organized into 5 groups (i.e. Configurations, Customization, Security, Language, and Reporting). Each option 112 is a hyperlink that will update the page 100, wherein the options panel 110 will remain unchanged and the details panel 120 will change in accordance to the selected option 112.

The details panel 120 in the present figure corresponds to the Directories option 112 (i.e. the "Directories panel"). Using this panel 120, an administrator may add new directory references to a directory shell, change properties of a existing

directory reference in the directory shell, or enable/disable directory references. The Directories panel 120 provides a table 122 listing various directory references in the directory shell. In the present example, each row of the table 122 corresponds to a DirectoryScheme object. The Configuration column in the table 122 corresponds to
5   the name of the DirectoryScheme objects. The Login Server column provides a radio button for each row to identify the server to which the directory shell authenticates. In the present example, only one server may be selected, however, it is contemplated that the directory shell could authenticate to more than one server.

The Authentication Group column of the table 122 provides a check box for
10   each row. The Authentication Group functionality breaks one large directory down into multiple, smaller directories for quicker search results. For example, each DirectoryScheme object could point to a different search base in the same directory, thus utilizing a multi-threaded search capability, such as that found in Novell's eGuide. Setting a DirectoryScheme object as a member of the Authentication Group
15   designates that directory to be searched with the user's authenticated credentials verses its default proxy credentials. Any directory residing outside of the Authentication Group will consume proxy credentials on searches. Preferably, a directory would be designated as a member of the Authentication Group only when the distinguished name ("DN") and passwords of the user are applicable in both the
20   Login Server and the intended directory. A user will have the ability to modify "Editable" attributes as long as they reside on the Login Server or on a directory that is an Authentication Group member.

One advantage of the Authentication Group functionality is that it utilizes a user's credentials for LDAP queries (inside the Authentication Group), thus utilizing
25   the Access Control List ("ACL"). In regard to consuming the ACL, critical information can easily be protected and controlled through directory rights. Directories containing "general" or "non-critical" information can simply be excluded from the Authentication Group, where proxy credentials will be sufficient for viewing its data. This poses many different powerful business-to-business solutions.

30   The Enabled column of the table 122 provides a check box for each row, thus allowing an administrator to disable or enable searching on a directory. The Properties column provides a link to enable the administrator to edit the DirectoryScheme object. Upon selection of the link, the page 100 will be updated to present the Directories Edit panel 130 (shown in Fig. 5, discussed below). The

Remove column in the table 122 allows an administrator to remove an DirectoryScheme object from the directory shell by selecting "X" in the associated row. Preferably, a user is prevented from removing the login server.

5   The Directories panel 120 includes two buttons 124. The Save button saves an changes entered in the table 122. The New button will create a new instance of the DirectoryScheme class, thus adding a new directory reference to the directory shell. Upon selection, the page 100 will be updated to display the Directories Edit panel 130. After creation, this new DirectoryScheme object will be added to the table 122.

10   Turning now to Fig. 5, the page 100 comprises an example of a Directories Edit panel 130. The panel 130 includes three tabs 132 (i.e. LDAP Settings, Attributes, and Advanced), each corresponding to a separate view of the panel 130. The default view for the Directories Edit panel 130 is the LDAP Settings view, as illustrated here. The LDAP Settings view comprises a variety of fields 134 relating to

15   the configuration of the directory that a DirectoryScheme object represents. The Enabled field specifies if Directory is "Searchable." The Name field is used as an identifier for the DirectoryScheme object. If a new DirectoryScheme object is being created, the field will be editable. The Host Name field refers to the IP Address or DNS hostname of the directory server. The Port field refers to the port value for the

20   directory server. The Enable SSL field can be configured to consume SSL, but may require SSL initial configuration. The Secure Port field refers to the secure port value when SSL is enabled. The Search Root field refers to the directory root. The Search Subcontainers field provides three options: Base, One, and All. Base specifies that the scope of a search includes only the base distinguished name, One

25   specifies that the scope of a search includes only the entries one level below the base DN, and All specifies that the scope of a search includes the base DN and all entries at all levels beneath that base.

Continuing with the various fields 134, the Max Search Entries field sets the maximum number of search results returned on any given search in the directory

30   browser. The Proxy User Name field sets the search proxy DN, and the Proxy Password fields sets the search proxy password. By leaving the Proxy User Name and associated Proxy Password blank, anonymous credentials will be used on directory queries. The Authentication Group field specifies if the directory is a member of the Authentication Group. The Authentication User Name field only

applies to a Login Server and refers to the Authentication Proxy DN. This Authentication User will be responsible for searching and identifying fully DN's during contextless login based on the LOGINKEY mapped attribute. Therefore, the authentication proxy will have read access to all DN's and the LOGINKEY mapped

5   attribute in the Login Server. A "user not found" error will be encountered on login attempt if the Authentication User does not have read access to these specified fields. By leaving Authentication User Name and the associated password field blank, the directory shell will assume anonymous credentials on all contextless login attempts. Likewise, the fields Authentication Password and Authentication Search

10   Root only apply to Login Servers only, and refer respectively to the password in conjunction with Authentication User Name specified above and the container of where Authentication Credentials search is to begin.

The LDAP Settings view 130 includes two buttons 136. The Save button saves any changes made to the fields 134. The Refresh Schema button will merge

15   new schema information with existing settings for this directory.

Turning now to Fig. 6, an example of an Attributes view 140 corresponding to the Attributes tab 132 is displayed. This detailed panel 140 allows an administrator to select any defined ObjectScheme object from the Object Type pull-down field 142. A comprehensive list of category attributes for the directory populates the table 144.

20   From this table 144, an administrator may define mappings and settings for the AttributeScheme objects for the selected ObjectScheme object. The Enabled column in the table 144 selects whether the class attribute is useable, which simply means the attribute data of this class attribute will be returned from a search of the directory. A directory read will occur when the detailed panel is requested by clicking

25   on a search button in the directory browser. The Template Key column is the assigned key-name given to each class attribute. Since different directories potentially have different names for class attributes, the key-name is the mechanism for the directory shell to treat these differing class attributes the same across multiple directory types. The Searchable column designates an class attribute as

30   "Searchable" in the directory browser. The Editable column designates an class attribute as "Editable" from the directory browser.

Turning now to Fig. 7, an example of the Advanced view 150 corresponding to the Advanced tab 132 is displayed. From the Advanced view 150, an administrator has the option of editing existing categories, removing existing categories, or

creating new categories. Each ObjectScheme object or category is a collection of directory classes that have been grouped together under one category name. In other words, any given category can reference or represent one or more classes of objects in a directory. An administrator selects any defined ObjectScheme object

5 from the Object Type pull-down field 152. The Available list 154 is a dynamic list of all available directory classes from the selected directory. The administrator can select or group any of these available classes together into the Selected list 156 for the category identified in the field 152. Likewise, directory class in the Selected list 156 may be removed. In the present example, the category USER references and

10 represents Inetorgperson, Organizationalperson, and Person directory classes. When the Save button 158 is pressed, a table of combined directory class for this category will be updated to reflect the new collection of directory classes under this category. Selecting the New button allows for the creation of a new category.

Continuing with the options in the options panel 110, the Display Layout

15 option 112 allows an administrator to configure the display of one or more ObjectScheme objects. Selecting the Display Layout will present a new details panel of the page 100, which includes three tab views: Search Form, List Form, and Details Form. In a "Search Form" tab view, an administrator may specify the order of category attributes that have been specified as "Searchable" in the Attributes view

20 140 for the selected category. The item selected as the top of this list will be the default "Filter" selection from the directory browser when this category is selected. In a "List Form" tab view, an administrator may designate specific category attributes that will appear in the list panel of the directory browser. Preferably, only those attributes that have been previously designated as "Enabled" in the Attributes view

25 140 of this category will be available. In the "Details Form" tab view, an administrator may designate ordering of category attributes to be displayed in the details panel of the directory browser. All attributes tagged as "Enabled" from the Attributes view 140 of this category will show up in this list and be rendered according to the saved order value.

30 The Photos Properties option 112 configures the use of photographs in directory shell. The user may specify where image files are stores, such as on a directory or on a file system, the maximum image size, whether a photo agreement is required, and whether users are allowed to upload images.

The Search option 112 allows an administrator to specify the number of default search rows to start with and the number of search rows to be displayed when a search is requested from the directory browser. Further, an administrator can specify which search criteria will be displayed in the directory browser. Examples of search criteria include: contains, does not contain, does not end with, does not equal, does not start with, ends with, equals, and starts with.

The General option 112 allows an administrator to select from different user interface themes and specify the URL for the default "Home" link for the directory browser. A toggle allows authenticated users or User Administrators to go right into the Modify form when the details panel of the directory browser is requested and edit rights are ascertained. Further a toggle enables an authenticated user to modify her own "Editable" attributes in the directory browser. The Colors option 112 allows the administrator to choose a color scheme for the user interface.

The Administrator Roles option 112 configures administrative roles for users of the directory shell. For instance, user administrators can be assigned to modify access to all "Editable" attributes that belong to other directory users in the Authentication Group. When a self-administration option is enabled and the user is authenticated, users can modify their own "Editable" attributes. The user administrator always has rights to modify other user's "Editable" attributes regardless of whether self-administration is enabled or not. User administrators receive edit privileges on all category attributes that: 1) have attributes designated as "Editable" from the Attributes view 140, 2) reside in the Login Server or the Authentication Group, or 3) directory "write" privileges have been granted to the user in question. User administrators also have an additional edit-mode "Browse" feature to manage org-chart or manager information. This browse feature will show a browse icon next to the manager DN attribute of all Authentication Group USERS, which allow manager searching and convenient insertion of the matching DN in the Manager field.

The Restrictions option 112 configures whether validated credentials will be stored on the client machine in a cookie using Base64 encryption. Subsequent loading of the directory shell will automatically authenticate the client with the credentials stored in the cookie. An administrator can also specify how many seconds until the cookie (if enabled and exists) will expire. Further, an administrator

can specify whether directory authentication is required prior the directory browser being available.

The Labels option 112 enables the administrator to configure the text of labels for AttributeScheme objects that will be presented in the list panel and details panel of the directory browser. Instead of presenting a somewhat cryptic class attribute name found in the directory, the administrator can configure the label to be more recognizable to a user. For instance, a class attribute may be "CELLPHONEEMAIL" and the corresponding text label may be "Cell eMail."

The Debug option 112 allows the administrator to enable diagnostic information to be written to a log file associate with a session, as well as configure the type of debugging information that the administrator desires to be written.

Fig. 8 illustrates an example of a page associated with a directory browser or client. As shown in this example, the page 200 generally comprises three portions: a query portion 210, an list panel 220, and a details panel 230. The query portion 210 comprises a variety of input options to format a query. The category input 211 in this example is a pull-down list of the various ObjectScheme objects, thus allowing a user to select a category or search object against which a query is directed. The attribute input 212 in this example is a pull-down list of the various AttributeScheme objects that have been identified as "Searchable" in the Search Form tab of the Display Layout option 112. Accordingly, a user may select a category attribute against which a query is directed. The criteria input 213 in this example is a pull-down list of different search criteria, which conform to the criteria selected under the Search option 112. The data input 214 in this example is an edit field that can accept alphanumeric string (sometimes referred to as query data) that will be the subject of the query. In the present example, the query format based on the values entered in the various fields 211, 212, 213, and 214 is a query against the "Last Name" category attribute of the "Find People" category that starts with the character "g". The Boolean expansion button 216, shown here as a "+" allows the user to format a Boolean query, which is described in greater detail below.

The search button 215 will format and submit a search request to the directories in accordance the query format provided by the user in the various inputs 211, 212, 213, and 214. For instance the search request to the directory will be directed to the classes of objects and associated attributes that correspond to the respective category and category attributes elected in the inputs 211 and 212. In the

present example, query format is converted into a directory packet request, such as an LDAP packet request, which will preferably be multi-threaded. The packet request is submitted to the appropriate directories, which will respond with a return packet with directory data. The return packet is converted into XML format, and then the XML data is rendered with an XSL style sheet to create WML or HTML pages depending upon the device. The new HTML page is then served to the user's HTML browser to populate the list panel 220 and the details panel 230 with the search results.

The list panel 220 presents a listing of the various directory objects satisfying the query. The presentation format of each object is configured in accordance with the administrator's selection in the List Form tab view of the Display Layout option 112. In the present example, four directory objects satisfied the query and the listed category attributes are First Name, Last Name, and Department. The details panel 230 lists a more detailed information regarding the directory object selected in the list panel 220. The category attributes and data listed in the details panel 230 are configured in the Details Form tab of the Display Layout option 112. Upon selecting the Edit link 232, assuming the user has rights to edit values in the directory, a modify screen will be present, such as that depicted in Fig. 9

Fig. 9 depicts another example of the directory browser page 200. In this example, the details panel 230 is presented as a Modify form. For instance, the value associated with the AttributeScheme object Preferred Name is presented in an edit field 234. The Modify form is presented when the user has appropriate right to modify value in the directory. Further, edit fields 234 are presented when the corresponding AttributeScheme object has been configured as "Editable". Upon selecting the Make Changes button 236 , the changes to the values are modified in the corresponding direct Attributes in the directory. The reset button 238 returns the edit field 234 to the current value in the directory.

Fig. 10 illustrates another example of the directory browser page 200. In the present example, the details panel 230 displays the objects selected from the list panel 220 in the form of a graphical organizational chart 240. Optionally, an Organization Chart tab 242 is also presented so that the user can toggle between the chart view and the view provided under the Information tab, as shown in Figs 8 and 9.

Fig. 11 illustrates another example of the directory browser page 200. In the present example, the query portion 210 has been expanded to format Boolean queries. The Boolean inputs 218 in this example are pull-down lists containing standard Boolean operators. The Boolean expansion button 216, shown here as a "-
5   " may be selected to eliminate the Boolean format.

The foregoing description and examples of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive nor to limit the invention to the precise form disclosed. Many alternatives, modifications, and variations will be apparent to those skilled in the art in light of the above
10   teaching. Accordingly, this invention is intended to embrace all alternatives, modifications, and variations that fall within the spirit and broad scope of the amended claims.